M10-L1 Problem 2: Solution

In this problem you will use the `sklearn` implementation of the K-Means algorithm to cluster the same two datasets from problem 1.

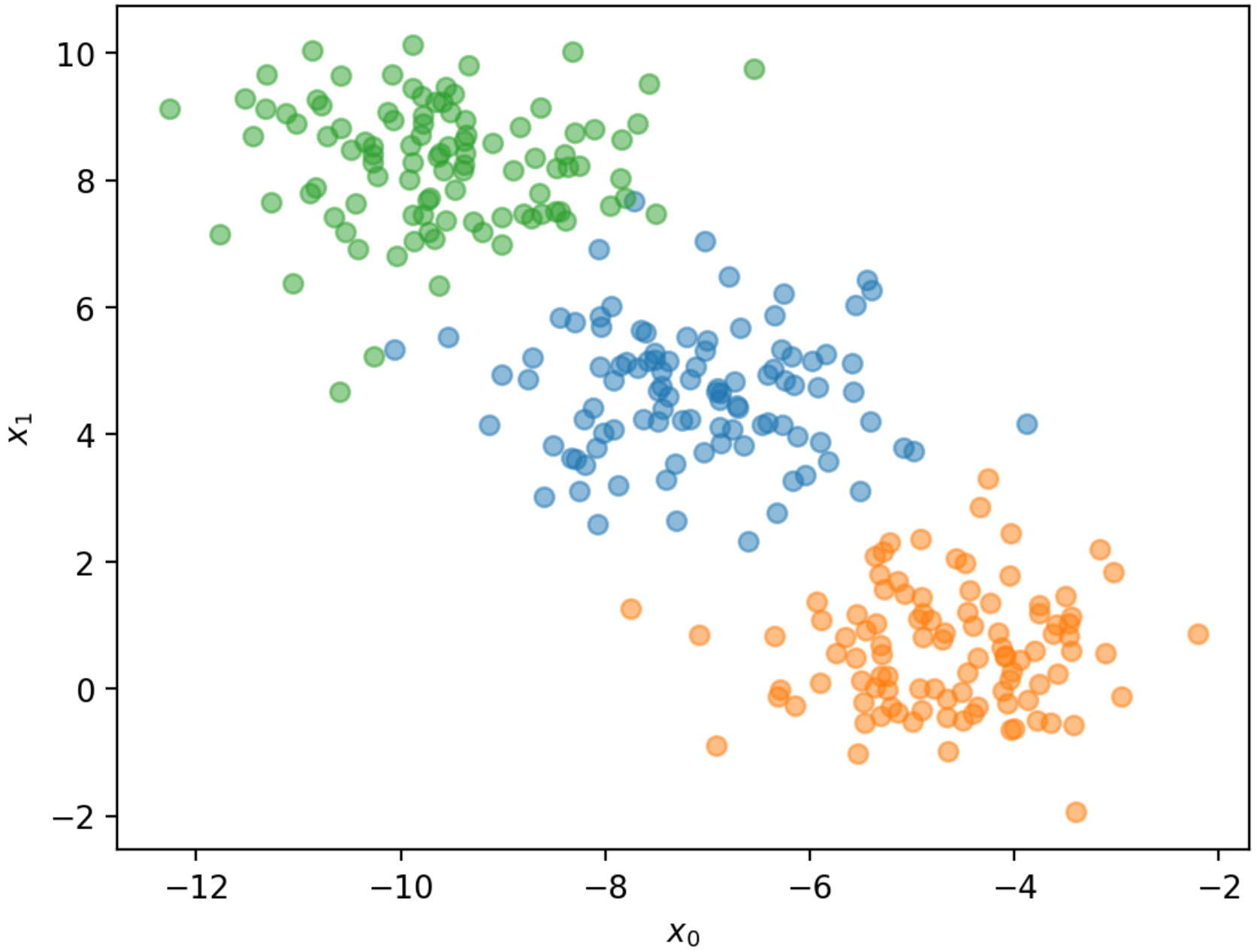
In [7]:
```
import numpy as np
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')

from sklearn.datasets import make_blobs, make_moons
from sklearn.cluster import KMeans

## DO NOT MODIFY
def plotter(x, y, labels = None, centers = None):
    fig = plt.figure(dpi = 200)
    for i in range(len(np.unique(y))):
        if labels is not None:
            plt.scatter(x[labels == i, 0], x[labels == i, 1], alpha = 0.5)
        else:
            plt.scatter(x[y == i, 0], x[y == i, 1], alpha = 0.5)
    if labels is not None:
        if (labels != y).any():
            plt.scatter(x[labels != y, 0], x[labels != y, 1], s = 100, c = 'None', edgecolors = 'black', label = 'Miscl
    if centers is not None:
        plt.scatter(centers[:,0], centers[:,1], c = 'red', label = 'Cluster Centers')
    plt.xlabel('$x_0$')
    plt.ylabel('$x_1$')
    if labels is not None or centers is not None:
        plt.legend()
    plt.show()
```

We will use `sklearn.datasets.make_blobs()` to generate the dataset. The `random_state = 12` argument is used to ensure all students have the same data.

In [8]:
```
## DO NOT MODIFY
x, y = make_blobs(n_samples = 300, n_features = 2, random_state = 12)
```

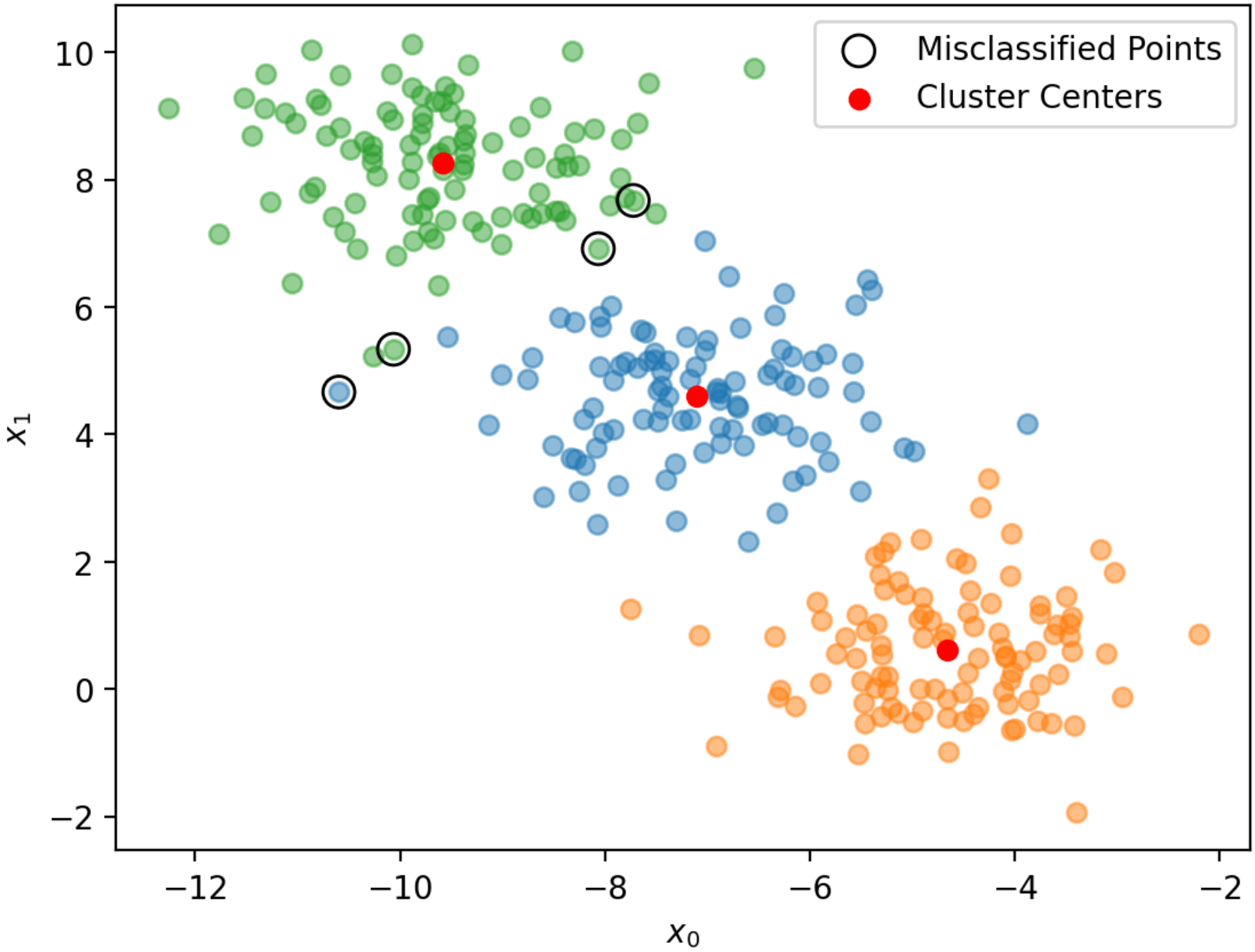
Visualize the data using the `plotter(x,y)` function. You do not need to pass the `labels` or `centers` arguments

In [9]:
```
## YOUR CODE GOES HERE
plotter(x,y)
```



Now you will use `sklearn.cluster.KMeans()` to cluster the provided data points `x`. For the `KMeans()` function to perform identically to our implementation, we need to provide the same initial clusters with the `init` argument. The cluster centers should be initialized as `np.array([[-5,5],[0,0],[-10,10]])`, and you can additionally pass in the `n_init = 1` argument to silence a runtime warning that comes from passing explicit initial cluster centers. Then plot the results using the provided `plotter(x,y,labels,centers)` function.

In [10]:
```
## YOUR CODE GOES HERE
init_centers = np.array([[-5,5],[0,0],[-10,10]])
kmeans = KMeans(n_clusters = 3, init = init_centers, n_init = 1)
labels = kmeans.fit_predict(x)
centers = kmeans.cluster_centers_
plotter(x,y,labels,centers)
```



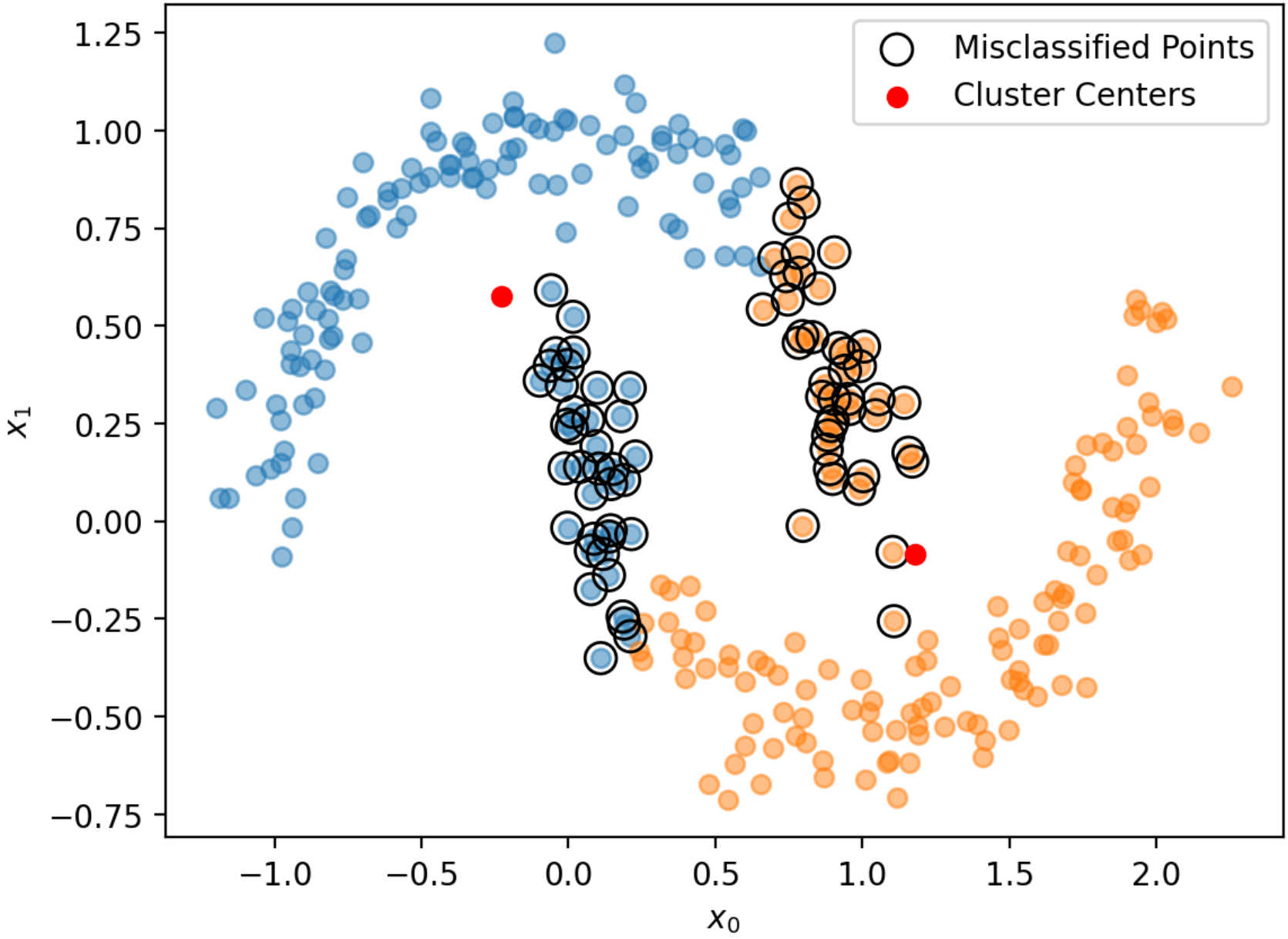
Moon Dataset

Now we will try using the `sklearn.cluster.KMeans()` function on the moons dataset from problem 1.

In [11]:
```
## DO NOT MODIFY
x,y = make_moons(n_samples = 300, noise = 0.1, random_state = 0)
```

Using the same initial cluster centers from problem 1, namely, `np.array([[0,1],[1,-0.5]])`, cluster the moons datasets and plot the results using the provided `plotter(x,y,labels,centers)` function.

In [12]:
```
## YOUR CODE GOES HERE
init_centers = np.array([[0,1],[1,-0.5]])
kmeans = KMeans(n_clusters = 2, init = init_centers, n_init = 1)
labels = kmeans.fit_predict(x)
centers = kmeans.cluster_centers_
plotter(x,y,labels,centers)
```


Misclassified Points
Cluster Centers
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
x_1
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x_0

Discussion

How do the results of your hand coded implementation of the K-Means algorithm compare to the `sklearn` implementation? If there is any discrepancy between the results, provide your reasoning why.

Both the codes provied the same results in clustering. As both the graphs were completely similar to each other, there was no discrepancy.

In [ ]: